# OMSC Army Unit Abstract Object Class

(Revised: 1 June 1999)

The OMSC designed a Unit Object Class for use in Army models and simulations (M&S). The development of this class is based on a component approach to class development. This component approach supports the use of both object composition and class inheritance methodologies. The resulting class is composable in nature. This allows each M&S developer to tailor the object to their specific application while maintaining a standard model structure and a minimum set of common model interfaces. The Unit Object is organized along the following lines:

- Unit Superclass
  - Unit Component Class
    - Unit Geometry Class
    - System Group Class
      - Platform Class
      - PlatformInfo Class
    - Logistics Class
      - Supply Subclass
      - Maintenance Subclass
    - C2 Class
    - Attrition Class
    - Communication Class
    - Intel Class

The attached figure outlines the structure of the Unit Object. Also provided are definitions for each of the components of the Unit Object class as well as descriptions of the public methods associated with each class.

## Unit Object Class and Component Definitions

**Class Unit**: A "Unit" is any military organization that is composed of multiple entities. Examples include military organizations such as a company, battalion, brigade or division.
Public Methods:

getLocation(): Returns the current unit location. Typically this is the center of mass or some other point location representative of the unit location.
getVelocity(): Returns the current velocity (direction of movement and rate) of the unit.
getMvmtDirection(): Returns the movement direction for a unit moving from one location to another.
getID(): Returns a string that identifies the unit.
getSide(): Returns the faction or coalition for the platform. There is no implied enmity between sides.
getPosture(): Returns the unit posture. Examples of posture might be operational activities like road march, hasty attack, hasty defense, etc.
getStatus(): Returns the unit status. Status is used for planning. Examples might include a percent effectiveness (based on system weights), fraction on hand (number on hand divided by number authorized), unit effectiveness state (an enumerated type based on the percent effectiveness), relationship with objective (an enumerated type based on distance to current objective). There may also be fuel and weapon statuses and a status based on enemy fire.
getMission(): Returns the unit mission. An example is the current task the unit was ordered to accomplish.
getEchelon (): Returns the unit echelon. Examples are battalion, brigade and division.
move(): Used to advances a unit toward its next location.
determineAttrition(): Used to calculate the attrition caused by another unit or platform.

**Class UnitComponent**: A "Unit" is partitioned into logical components so that the modeler can compose a unit from various components. Components may be extended through inheritance. All of the components listed below will inherit the following method from this class.
Public Methods:

getStatus(): Returns the status of the unit or component.

**Class UnitGeometry**. The unit geometry describes the shape or foot print of the unit on the ground, the layout of systems within the unit, the unit search area, and unit orientation and posture. Geometry may be used for attrition, sensing and movement.
Public Methods:

getShape(): Returns the bounding shape of the unit.
getOrientation(): Returns the general orientation of the systems within the unit location.

## Unit Object Class and Component Definitions
(continued)

**Class SystemGroup**. This component accounts for individual systems (or platforms) within the unit.

Public Methods:

getQty (): Return the number of systems of this type in the unit.
acceptLosses (): Used to decrement the number of systems of this type in the unit.
acceptGains (): Used to increment the number of systems of this type in the unit.

**Class Platform**. A platform can be any entity of interest in the model. Examples include vehicles of all types, individuals/persons, individual systems (i.e., radar systems), a missile, etc. The complete definition for this class is provided separately.

**Class PlatformInfo**: This component contains static information and/or data about the various platforms contained within the unit. Examples include the gross weight of a vehicle, a description of the size or type of weapons mounted on the platform, etc.

**Class Logistics**. This component is intended to capture or represent the internal logistics capability and/or requirements of the unit. This covers both supply and maintenance requirements and/or activities.

Public Methods:

receive(): Used to increment the quantity of this logistic component.

**Class Supply**. A supply component of a unit such as ammunition class.
Derived from Logistics.

Public Methods:

getReminaingCapacity(): Returns the remaining capacity for this supply component.
getTotalCapacity(): Returns the total capacity for this supply component.
transfer(): Used to transfer a quantity of an on hand supply component to another unit or platform.

**Class Maintenance**: A maintenance component of a unit such as a repair action.
Derived from Logistics.

Public Methods:

conductMaintenance(): Used to perform maintenance actions on equipment and medical treatment for individuals.
conductRecovery(): Used to recover items from an area of operations.
conductEvacuation(): Used to evacuate equipment and/or individuals to rear areas.

**Unit Object Class and Component Definitions**
(continued)

**Class C2**. This component is used for command and control decision making in the unit. A unit may have more than one command and control component (itself, subordinate units, etc.).
Public Methods:

doC2(): Used to initiate a command and control cycle where command decisions are made and control actions initiated.

**Class Attrition**. The attrition component allows the unit to cause loses to another unit. This is a separate class because damage can be inflicted in many ways (e.g., direct vs. indirect fire).
Public Methods:

CauseAttrition(): Used for the unit to cause losses to another unit.

**Class Communications**: This component provides the ability to explicitly model communications.
Public Methods:

getNet(): Returns the collection of objects capable of exchanging messages.
setNet(): Used to add the unit to the collection of objects capable of exchanging messages.
sendMessage(): Used to send a message on the net.
receiveMessage(): Used to receive a message from the net.

**Class Intel**: This component provides the ability to explicitly model the intelligence collection process performed by units with their organic sensor assets.
Public Methods:

collect(): Used to initiate local detection using the unit search capabilities.
reportContacts(): Used to report the results of the collect() operation.